**A Database Manager and Creator**

*Aidan MacKenzue*

*CSCI 24000*

**Project Summary**

**Purpose**

The purpose of this program is to allow multiple users to create Databases comprised of an entry and a description for whatever the user decides.

**Description of Program**

THis program works with serialization and file IO to allow for multiple users to have unique files and unique data between databases and the entries in said databases. The program is managed by an admin account who can check in on users to see if their data is harmful. The admin works to weed out bad actors in the system and keep the users aligned with a theoretical “terms of service”. THe Users are allowed to create as many databases as they desire and as many entries in their databases as they desire. Each user is added to an overarching arrayList tied to a file. The arraylist is of type class. The class being the account which is tied to the user’s username and password. This helps to keep all data members and class data personal to the user and discourage bugs such as one user having their data able to be accessed by other users. By serializing the entire account, all users have a personal menu tied to their account.

**Description of Classes**

1. Main Class

The Main class handles all of the workings of account creation and logging in. Upon creation of a new account. The set username and password is saved to the serializable account class and the entire class and sub classes of account are saved to thart specific user inside of a file designed to hold the user’s username, password, and account.

1. The Admin Class

The admin class acts as an overseer for the users in the system. The admin can view all users, delete a user from the system, and even flag user data as hostile and delete said data and warn the user of their malicious deeds not being tolerated

1. Account Class

The account class holds the menu for account actions pertaining to the creation of databases. Each account class is tied to a specific user via their username and password. THis helps to keep all data original. From here the user can delete a database, create a database, access a database, or log out of the system. THis class also contains a secret method that the admin access in order to check up on the user. THis class also uses an arraylist of type Database to tie the Database class to the dbName datamember. THis class also contains a a constructor to set the original username and password as well as getters and setters for such

1. Database Class

A subclass of the account class. This class, much like account class. This class is also tied to a datamember. This class is tied to the datamember for the name of the database to protect the originality of the data stored here. The user has basic CRUD options as is typical of a database. THere is another secret method here to allow for the admin to access this system and flag data as malicious. THis class contains an arraylist of type Entry to enter data and save data. THis class also contains a constructor to set the dbName for originality.

1. Entry CLass

This class is used to set and get data and uses a constructor to set entry and description values as well as getters and setters.

**Use Case Analysis**

**Admin Use Case**

* Admin logs in using special admin username and password
  + Admin has a choice between 1 of 4 menus
    - View Users in System
    - Delete User in System
    - View Data of Users in System
    - Log out
    - Data from file with User Accounts is loaded in at this time into an ArrayList
  + Admin decides to first view all users
    - Program will take the arraylist of User Accounts and spit out the username of the users and the password for those users along with an index value of their place in the system
  + Admin then decides to view the data of one of the users
    - The program will spit out just the username and the index value of the users
    - Admin will select the index value or be allowed to quit out of the system
    - Admin selects a user
      * Admin is taken to a “Secret” menu unavailable to the user in their Account
      * Admin can select one of the user DB’s to access.
      * Admin Selects a DB o View
        + Admin is then accessing another Secret method of the Database class and all of the user’s entries are displayed
        + Admin can then decide if they wish to flag an entry

IF an entry is flagged

Entry is deleted from system

New Entry is created warning the user as to why their entry no longer exists

Admin is returned to the menu

* + - * + Admin Logs out

Admin is returned to the login page

**User One Use Case**

* User Creates an account in the system
  + Account is stored in an ArrayList and saved to a file with all other users
* User logs into their account
  + The program will try to create a file with the username being the file name.
    - If the file already exists, then the program will enter the user’s menu without problems
    - The Account class is serialized so that all menus and data of the class is specific to each user to prevent mutliple users accessing the same data
    - User is then prompted with 4 options
      * Create DB
      * Access DB
      * Delete DB
      * Log out
      * Username fiile is loaded into an arraylist of type Database class which is serializable
    - User creates a DB
      * User is prompted to name their DB
    - User selects a name and that name is then saved into the arraylist
      * The Arraylist saves said information by setting the name in the serializable database class to the name chosen by the user
      * The ArrayList is thens aved into the username file
    - USer decides to open their new database
      * USer psi prompted with 5 choices
        + Create entry
        + View entry
        + Delete entry
        + Add entry
        + Edit entry
        + Log out
        + The arraylist of entries is loaded wirth all values in the DB name file
      * The Progam takes the index of the arraylist and spits out the index value and the name of the DB
      * USer selects their new db
      * Each db is opened through the ArrayList to prevent dbs having the same entries copied over. Each db is its own instance of the database class
      * A file is then created with the name of the db as the file name
        + If the file is already created, then the program will skip this step
    - User decides to create an entry
      * User is prompted to create a title for the entry
      * USer is prompted to create a description for the entry
      * Entry is added to the Serializable entries File by adding them to the constructor with parameters of String and String
    - USer decides to view their entry
      * Entries are printed out in a for loop with their index value, a header describing what is the entry and the description, and the physical data of the entry and description until the value of the for loop reaches the size of the arraylist
    - User notices a mistake in their entry and decides to edit it
      * User is given the entry tites along with their corresponding index value in the file via a for loop.
    - User selects the entry they wish to edit
      * USer is prompted to set a new description for the entry
        + Entry is saved without entry title being replaced via the arraylist method of set().
    - USer decides to delete their new entry
      * USer is given the entry tiles along with their index via another for loop
      * Selects entry to delete
        + User is prompted to confirm their delete
      * USer confirms delete
        + Item successfully deleted and the arraylist and file are saved as such
    - User logs out of database
      * User is returned to their menu
    - USer deletes database
      * USer is given all the index values of their DBs and the title of their dbs and asked to select which item to delete
    - USer selects item they wish to delete
      * USer is prompted to confirm their choice
        + Upon confirmation…

A try catch is initiated to see if the value they inputted is legal, and if that item can be used to delete the value

If successful

Item is deleted and user is taken back to their menu and arraylist and file is saved

If unsuccessful

Exception handler claims something went wrong

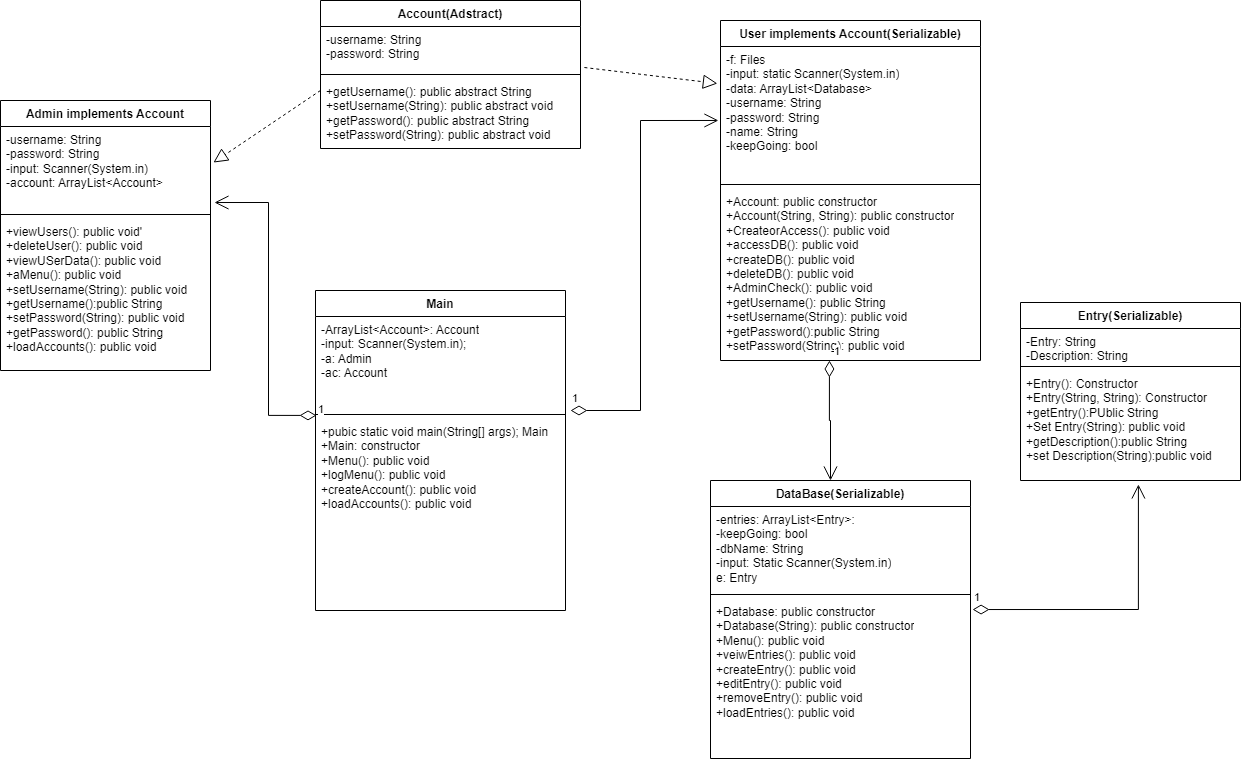
* + - USer logs out

**Course Ideas Used**

1. Encapsulation
   1. All methods of the account and database classes are hidden and only accessible with the proper input of a valid username and password
2. Composition
   1. Each User has their own instance of the account class
   2. Each database has their own instance of the Database and entry class
3. Aggregation
   1. Main and Admin each hold ArrayLists tied to the Account class methods
   2. The Account class instances have an arraylist of Database class methods
   3. Each Database class has an arraylist of Enty class methods
4. Language
   1. IDeally this program would be written in Java for its easy use of FIle IO and ArrayList. Java includes object serialization easily. Program uses a Command Line Interface
5. Abstraction
   1. Both the admin and the User class inherit from the abstract class of Account. This class contains the string of username and password as well as getters and setters for each.

**Data Design**

UML Diagram



**UI Design**

This program is written in the Command Line and will use The Java Language. The UI Will be set uf for the user to input their commands in the form of numbers(Taken as a string) that coorespons to different if statements. These numbers will has a short sentence or few words describing what their action does

EXAMPLE

1. **Create Entry**
2. **Edit Entry**
3. **Delete Entry**
4. **Log Out**

**Algorithm**

* Admin
  + Non Method class items
    - Variables to include
      * String for username
      * String for password
    - Other includes
      * A Scanner for input that has a System.in
      * An ArrayList
        + ArrayList has type Account

Name it accounts

* + Implements Account class
  + Include a method for setting the username
    - Method has a parameter of string
    - Set the parameter to what ever desired admin username is
    - Set the this.username to the value of choice
  + Include a method for getting the username
    - Method has type string
    - Method returns the username string
  + Include a method for setting the password
    - Method has a parameter of string
    - Set the parameter to what ever desired admin password is
    - Set the this.username to the value of choice
  + Include a method for getting the password
    - Method has type string
    - Method returns the password string
  + Include a method for menu
    - Menu will output a list of options for the user to select from
      * View users
      * Delete User
      * Access USer Data/Delete user data
      * Log out
    - User can select one of these values with a string input and will achieve the desired outcome from the input
    - Program will loop with misinputs
  + Include a method for viewing all users
    - A for loop is created to list all user’s username and password along with their index value in the system
  + Include a method for deleting a user
    - A for loop lists all usernames and their index value in the system
      * The Admin enters a string value of the index selected
      * Admin is prompted to confirm their choice
        + Upon confirm

Admin’s string value is converted to an integer

If this is not possible, a catch will catch the exception and keep the program running

Integer value deletes the user in question

* + Include a method for viewing user data and deleting if necessary.
    - A for loop lists all usernames and their index value in the system
      * The Admin enters a string value of the index selected
      * The admin is sent to the user class associated with the index value selected
        + Sent to a method inaccessible to the user
* Main
  + Non method Includes
    - A Scanner for input that gets System.in
    - A boolean called keepGoing
    - An ArrayList of type User
      * Called accounts
    - Admin class
      * Called a.
    - Public static void main(String[] args)
  + Constructor Main
    - Calls the method for menu
  + Method for menu
    - Menu gives 2 options
      * Options are to create an account or log in
        + Each take to to the respective page
  + Method for create account
    - User is prompted to input their new username
      * User gives their username with the input scanner
    - User is prompted to enter a password
      * User inputs their password
    - User confirms password
      * Prompted to input the password again
        + If passwords match

Values are added to the array list through class constructor.

A try catch is opened for a object output stream

The arraylist is then written to a file

* + - * + If passwords dont match

Program returns to log in or create account

* + Method for login menu
    - User is prompted to input their username and password
      * User inputs their username and password
        + An if else is opened to see if there is a file with the user’s username in existence

A file is opened if no

A file is not opened if yess

The user is taken to their specific menu via a for loop to find matching username and password

* + Method for loading the accounts
    - A object input stream is opened in a try catch
      * The arraylist is then opened and rewritten with the data in the file. This is used to load the data in menu
* Account
  + Non Method Includes
    - String for username
    - String for password
  + Abstract class
    - Implemented by User and admin
  + Include a method for getting the username
    - Method has type string
  + Include a method for setting the username
    - Method has parameter string
  + Include a method for getting the password
    - Method has type string
  + Include a method for setting the password
    - Method has parameter string
* User
  + Implements account and is serializable
  + Non Method inclusions
    - String for username
    - String for password
    - String for name
    - Boolean keepGoing
    - A scanner input for System.in
    - An ArrayList with Type Database
  + Constructor with no parameters
    - Sets the username and password to empty strings
  + COnstructor with 2 string parameters
    - These parameters are what the empty strings are set to
  + Method for loading the databases.
    - This method will ber a try catch and have an object input stream to load all the DBs
  + Method for menu
    - Gives the user a list of options and and a number value associated with the options
    - The user then inputs a string value and loops if the value is not equal to a valid option
  + Method for adding a db
    - USer inputs a name for the db
    - Name is sent the the parameter constructor of the database class and saved to ana rray list
    - Class is serialized in the arraylist with an object output stream and try catch
  + Delete a db
    - User is given the names of the db’s and the index value associated with such
    - User is prompted to select a db to delete
    - USr is prompted to confirm delete with an ther input string
    - DB is deleted if user inputs legal value
    - All done with string inputs and integer parsing
  + Access a db Method
    - User is given the names of the db’s and the index value associated with such
    - Program is prompted to create a file with the name of the file being the name of the db
    - If such a file exists . the file is not created
  + INclude method for adminChecking
    - Once an admin selects this class to view
      * A for loop prints out the databases associated with this user
    - Admin then is prompted to select a database
      * Input scanner gets an integer value
    - Integer value is used to prompt the arraylist to send the user to that db’s menu
* Database
  + Implements serializable
  + Not method includes
    - Boolean keepGoing
    - AN Arraylist of type entry
    - A string for database name
    - An Entry class instantiation
  + Constructor with no parameters
    - Sets database name string to an empty string
  + Constructor with parameters of string
    - The emty string is set to the string in the parameter
  + Method for loading the entries
    - A try catch is used for an object input stream
      * Arraylist is loaded with data inside the file
  + Method for menu
    - Menu lists numbers associated with options for the user
    - USer can select to add, view , edit, and delete entries along with the ability to log out of the system
    - load Entries is also called at this time
  + Method for View Entry
    - A for loop lists all the entries and their description
  + Method for Add Entry
    - The user is prompted to add a title for their entry
      * The user gives a title with a string input
    - User is prompted to give a description for their entry
      * The user gives a description for their entry with a string input
    - The entry is saved to the arraylist via the 2 string parameter constructor in the entry class
  + Method for Edit entry
    - A for loop lists all the entry titles
      * User gives the index value associated with their entry of choice
        + The Entry title is saved to a variable via the getEntry method of the entry class
      * The user is prompted to give a new description for the entry
        + User inputs a new description
      * The user sets the entry with the entry title to the new description value
  + Method for delete entry
    - A for loop lists all the entry titles
      * User gives the index value associated with their entry of choice
    - The user is prompted with a choice to confirm their entry
      * USer confirms entry with a string input
      * Entry is converted into an integer value and a remove method deletes the value from the arraylist
  + Method for admin access
    - Admin is given the values in the DB with the admin acces upon selecting a DB
    - Admin decides if they want to flag entries.
    - Admin flags entry
    - Entry is deleted from system along with a new entry replacing the deleted on informing the user that their entry was flagged and has been deleted
* Entry
  + Used by the Database class to save data
  + Includes a string for Entry title
  + Includes a string for entry description
  + Includes a constructor
    - Constructor sets the name and description variable to empty strings
  + Includes a parameter constructor
    - Parameters are both variables for entry and description
    - Set both this.variables to get another variable
      * This will be used in the Database area
  + Include a method for getting the entry title
    - Method has type string
    - Method returns the title string
  + Includes a method for setting entry title
    - Has a parameter of a string
    - Set the this.variable to get the parameter
  + Include a method for setting the description
    - Method has a parameter of string
    - Set the this.variable to get the string parameter
  + Include a method for getting the description
    - Method has type string
    - Method returns description

**General Thoughts**

I made my project around data because I am interested in learning more about File IO and data handling. I set aside about a week for bug fixing as in past assignments I havent been able to work through bugs in an expedited manor. This proved to eb true as I was having issues with files not being specific to one user and a user’s filed being accessed by multiple users. The same issue was occurring with entries being copied over to multiple databases. This proved to be an issue with my design and was a relatively easy fix as the code was all there, but I needed to change how the design of my data worked without changing any code. I also had issues with the admin being able to access the entries and flag entries. Overall this program proved to be simple, but still challenging as I learned more about how File IO is managed and serialization when it comes to multiple users and multiple files.